



10-15-07

AF/2173
SFC

PTO/SB/21 (10-07)

Approved for use through 10/31/2007. OMB 0651-0031

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM (to be used for all correspondence after initial filing)	Application Number	09/911,663-Conf. #3836
	Filing Date	July 24, 2001
	First Named Inventor	John CIOLFI
	Art Unit	2173
	Examiner Name	N. Pillai
Total Number of Pages in This Submission	Attorney Docket Number	MWS-072

ENCLOSURES (Check all that apply)

<input type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment/Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Reply to Missing Parts/Incomplete Application <input type="checkbox"/> Reply to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Drawing(s) <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____ <input type="checkbox"/> Landscape Table on CD	<input type="checkbox"/> After Allowance Communication to TC <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input checked="" type="checkbox"/> Appeal Communication to TC (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input checked="" type="checkbox"/> Other Enclosure(s) (please identify below): Return Receipt Postcard Supplemental Appeal Brief
Remarks		

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	LAHIVE & COCKFIELD, LLP		
Signature			
Printed name	Kevin J. Canning		
Date	October 12, 2007	Reg. No.	35,470



Docket No.: MWS-072
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
John Ciolfi

Application No.: 09/911,663

Confirmation No.: 3836

Filed: July 24, 2001

Art Unit: 2173

For: HANDLING PARAMETERS IN BLOCK
DIAGRAM MODELING

Examiner: N. Pillai

SUPPLEMENTAL APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Appellant submits a supplemental appeal brief in response to the Notice of Non-Compliant Appeal Brief mailed September 12, 2007 and the Advisory Action mailed September 20, 2007.

In the supplemental appeal brief, the Status of Amendments section has been changed to identify the Response filed on November 8, 2006 and to add the status of pending claims and canceled claims, as suggested by the Examiner.

This brief contains items under the following headings as required by 37 C.F.R. § 41.37 and M.P.E.P. § 1205.02:

- I. Real Party in Interest
 - II Related Appeals and Interferences
 - III. Status of Claims
 - IV. Status of Amendments
 - V. Summary of Claimed Subject Matter
 - VI. Grounds of Rejection to be Reviewed on Appeal
 - VII. Argument
 - VIII. Claims Appendix
 - IX. Evidence Appendix
 - X. Related Proceedings Appendix
- Appendix A

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is:

The MathWorks, Inc.

II. RELATED APPEALS, INTERFERENCES, AND JUDICIAL PROCEEDINGS

Appellant appealed in August, 2005 from the final rejection mailed on February 17, 2005 (Paper No. 9). In response to the appeal, the prosecution was reopened in the Office Action mailed February 10, 2006 (Paper No. 4).

There are no other appeals, interferences, or judicial proceedings known to Appellant, the Appellant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application

There are 13 claims pending in application.

B. Current Status of Claims

1. Claims canceled: 1-32 and 45
2. Claims withdrawn from consideration but not canceled: N/A
3. Claims pending: 33-44 and 46
4. Claims allowed: N/A
5. Claims rejected: 16-24 and 26-46

C. Claims on Appeal

The claims on appeal are claims 33-44 and 46.

IV. STATUS OF AMENDMENTS

A Response to the final Office Action of September 8, 2006 was filed on November 8, 2006 without amending the claims. An amendment canceling claims 16-24, 26-32 and 45 was concurrently filed with an appeal brief on June 8, 2007. Claims 33-44 and 46 are pending and claims 1-32 and 45 are canceled.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter relates to a computer-based graphical environment for creating, modifying, executing (simulating) and/or generating code for models. The elements of a model may be presented as blocks, having

inputs, outputs and/or parameters that may influence the behavior of the blocks.

When a block receives data, it may perform operations on that data. Such operations are sometimes referred to as “block methods” that are associated with a block. For example, an output block method may be formally expressed as:

$$y = \text{OutputMethod}(t, x, u, p)$$

where “y” represents block output(s), “t” represents time, “x” represents zero or more states in the block or a model, “u” represents block input(s), and “p” represents a set of parameters supplied to the block. (see Background of the invention). A user may specify the parameter(s) **p** that are to be used by the block. The user-specified parameters may be in the form of numerical values, variables defined as constants, interfaced variables or some combination thereof. An interfaced variable is a variable whose value can be changed either during simulation or in generated code.

Models represented as block diagrams may be executed in an interpreted environment or they may be compiled. In addition, code can be generated from the block diagrams.

The claimed subject matter is directed to receiving a model and processing user-specified parameters for one or more blocks in order to convert them to run-time parameters. The run-time parameters are the block parameters that are used during execution of the block diagram model or in generated code. They are derived from user-specified parameters. The run-time block parameters may be produced from the user-defined block parameters in such a way as to improve or optimize resource allocation, such as the use of memory space during the execution of the model.

Some user-defined block parameters may be processed in such a way that the data for the corresponding run-time block parameters is reused across two or

more parameters from one or more blocks. In particular, like non-interfaced parameters may be “pooled” together – that is, a single common representation may be used for multiple parameters. Such a single representation may be, for example, an entry in a run-time parameter table. Generally, the parameter pooling process identifies block parameters having parameter data that matches a given criterion, and allocates only one copy of the parameter data for all references to a given parameter data set. The criterion may require that the data match bit-for-bit. Alternatively, the criterion can require that the data match exactly only after a mapping function has been applied to the data. A block may reconstruct the parameter data it needs for execution from the pooled parameters by using the parameter directly or using the mapping function during execution.

In some embodiments, blocks of the model may define a mapping function or a method that maps from the user-defined parameters to the run-time parameters. The user may enter parameter information while representing high-level abstractions, such as, for example, using equations or representing real-world value by using a floating point data type, and the user-defined parameter can be converted to a more efficient run-time data store representation, such as an integer data type.

Independent claim 33 is directed to a method of mapping graphical block parameters in a graphical block diagram modeling environment. User-defined block parameters are received, see, e.g., Specification, page 9, lines 14-18 and reference character 72 in Fig. 4 and processed to produce run-time block parameters, see, e.g., Specification, page 12, line 31 through page 13, line 12 and reference character 156 in Fig. 11. Like non-interfaced run-time block parameters are pooled together to reuse data for the like non-interfaced run-time block parameters. See, e.g., Specification, page 13, line 17-31 and reference character 168d in Fig. 12.

Independent claim 46 is directed to a medium for use in a graphical modeling environment on an electronic device. The medium holds instructions executable using the electronic device for mapping graphical block diagram block parameters. A plurality of user-defined block parameters is received, see, e.g., Specification, page 9, lines 14-18 and reference character 72 in Fig. 4, and processed to produce a plurality of run-time block parameters, see, e.g., Specification, page 12, line 31 through page 13, line 12 and reference character 156 in Fig. 11. Like non-interfaced run-time block parameters are pooled together to reuse data for the like non-interfaced run-time block parameters. See, e.g., Specification, page 13, line 17-31 and reference character 168d in Fig. 12.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 33-44 and 46 are rejected under U.S.C. §102(e) as being anticipated by United States Patent No. 6,937,257 B1 to Dunlavey (“Dunlavey”).

VII. ARGUMENT

Appellant believes that the following arguments address each of the grounds of rejection to be reviewed on appeal.

A. Claims 33-44 and 46 are not anticipated by Dunlavey because Dunlavey does not disclose “pooling together like non-interfaced run-time block parameters to reuse data for the like non-interfaced run-time block parameters.”

The Examiner has rejected Claims 33-44 and 46 under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,937,257 B1 (Dunlavey).” (See Office Action of September 8, 2006, page 2). To establish a *prima facie* case of anticipation, each and every element and limitation of the claimed invention must be disclosed expressly or inherently in a single prior art reference. *RCA Corp. v. Applied Digital Data Sys., Inc.*, 730 F.2d 1440, 1444, 221 USPQ 385, 388 (Fed.

Cir. 1984). Appellant respectfully submits that Dunlavey fails to disclose each and every element and limitation of claims 33-44 and 46 for the reasons set forth below.

Independent claim 33, which claims 34-44 depend upon, reads:

“A method of mapping graphical block diagram block parameters in a graphical block diagram modeling environment, comprising:
receiving a plurality of user-defined block parameters;
processing the plurality of user-defined block parameters to produce a plurality of run-time block parameters;
pooling together like non-interfaced run-time block parameters to reuse data for the like non-interfaced run-time block parameters.”

Independent claim 46 reads:

“A medium for use in a graphical modeling environment on an electronic device, the medium holding instructions executable using the electronic device for performing a method of mapping graphical block diagram block parameters, the method comprising:
receiving a plurality of user-defined block parameters;
processing the plurality of user-defined block parameters to produce a plurality of run-time block parameters; and
pooling together like non-interfaced run-time block parameters to reuse data for the like non-interfaced run-time block parameters.”

1. Dunlavey fails to disclose “pooling together like non-interfaced run-time block parameters”

Dunlavey relates to a graphical model editor for pharmacological computational models. A user of the editor may specify units for various variables

in the pharmacological model. As the user is entering the variables, the expressions representing the units of the variables are converted into an internal format. (Dunlavey, Abstract). The internal format is a single multidimensional data type, used for all variables. This multidimensional data type information is automatically propagated for each statement in the internal format to identify inconsistent units during the model creation. When such inconsistent units are identified, a warning message is generated to notify the user, substantially immediately after the inconsistent units are created. (Dunlavey, Abstract).

The multidimensional data type of Dunlavey is used exclusively during the model definition stage, and not during the execution stage, unlike the run-time parameters of the present Application, which are specifically defined as “parameters that are used during execution of the block diagram model or in generated code” (Detailed Description, page 9, lines 20-21). The Examiner asserts that Dunlavey teaches producing run-time parameters at column 3, lines 1-8, in describing the multidimensional unit type data. Appellant is puzzled by that assertion, because in those very lines Dunlavey explicitly states that the multidimensional unit type data is used “to identify inconsistent units while the model is being constructed” (Dunlavey, column 3, line 8). That is, the multidimensional data type cannot be said to be a run-time parameter, because it is used during model construction, not during model execution, as required by the present application.

Dunlavey is entirely unconcerned with how run-time parameters are created, and, as such, does not teach or suggest pooling together run-time parameters. No section of the discussion related to generating an executable model (Dunlavey, Figs. 5 and 6 and accompanying description) addresses creating run-time parameters from user-defined parameters or even the use of the multidimensional data type during model execution.

In comparison, claims 33 and 46 require “pooling together like non-interfaced run-time block parameters.” The Examiner refers to Dunlavey, Figure 4, SetDiscrete as disclosing the pooling aspect of the present application. (See Office Action of September 8, 2006, page 3). The SetDiscrete is listed in Figure 4 as one of the internal data structures of Dunlavey. It is not discussed anywhere in the text of the specification of Dunlavey, and, as such, Appellant is unclear on its meaning. However, as far as Appellant can discern, the SetDiscrete primitive is used to set a group of *categorical variables* that are jointly distributed (emphases added) (See Figure 4), where the term “distributed” refers not to distribution of variables in memory, but to the probability function distribution of the associated variables. For example, Dunlavey discloses BodyWeight and Age variables as the examples of the categorical variables, which may be set by the user to be interrelated (in the probability sense) and jointly distributed. (See Dunlavey, column 11, lines 29-39). Therefore, the SetDiscrete primitive of Dunlavey is merely an abstraction representing a mathematical relationship between variables, and not an indication that those variables are in any way pooled into a single parameter.

Furthermore, the Examiner’s interpretation of the SetDiscrete primitive of Dunlavey as a grouping of the run-time parameters contradicts the Examiner’s own assertion that the unit multidimensional data type of Dunlavey corresponds to the run-time parameters of the present application. Shown in Figure 4 are primitives used in the internal representation of Dunlavey. This internal representation is later converted to an executable model (see Dunlavey, column 19, lines 45-47). Figure 4 includes unit data type primitive (see Dunlavey, Figure 4, primitive “Unit”), as well as the primitives representing operations on it (see Dunlavey, Figure 4, primitives “TimesUnit” and “UnitPhrase”), which are explicitly stated to be operations on units. The primitive SetDiscrete is not linked or indicated to be related to any of the unit primitives, as it would be, if

SetDiscrete represented grouping of the run-time variables and if the multidimensional unit type represented the run-time variables. Therefore, the Examiner's interpretation of Dunlavey is inconsistent.

Not only does Dunlavey not teach or suggest pooling of run-time parameters, it teaches away from having "non-interfaced" run-time parameters – that is, parameters that are not modified during the execution. In discussing the simulation of the model in connection with Figure 7C, Dunlavey states that "the values for the various parameters can be changed by entering new values in the value field of the parameters portion 750..." (see Dunlavey, column 25, lines 40-42). This statement suggests that all parameters may be changed by the user during the execution, unlike the non-interfaced parameters of the present application, which are not modified by the user during the execution.

Appellant therefore submits that Dunlavey does not disclose "pooling of like non-interfaced run-time parameters," as recited by claims 33 and 46.

In the "Response to Arguments" section of the Office Action of September 8, 2006, the Examiner alleges that "Dunlavey does disclose examples of when like parameters are grouped or pooled together (column 24, lines 35-45)." (See Office Action of September 8, 2006, page 10). Appellant respectfully disagrees and contends that Dunlavey does not disclose "examples of when like parameters are grouped or pooled together," as alleged by the Examiner.

The section of Dunlavey referenced by the Examiner recites that "all of the generated internal format statements are sorted into dependency order." (See Dunlavey, column 24, lines 35-36) What is disclosed in the referenced section of Dunlavey is the process of determining the execution sequence of the generated internal format statements. No interpretation of Dunlavey allows the internal format statements to be equated to the run-time parameters. The internal format statements represent operations performed on variables (see Dunlavey, column 24,

lines 40-49). Even if the variables used in Dunlavey were to correspond to the run-time parameters, no grouping of statements using the variables can anticipate grouping of the parameters themselves. Furthermore, the pending claims specifically recite “pooling” the parameters, not “grouping” them, as the Examiner seems to be implying. The sorting of the generated internal format statements is not “pooling of like non-interfaced run-time parameters,” as required in claims 33 and 46, but merely determining the execution sequence of the internal format statements. The Examiner’s allegation that Dunlavey discloses “the examples of when like parameters are grouped or pooled together” is simply misplaced. Dunlavey does not disclose “pooling of like non-interfaced run-time parameters,” as required in claims 33 and 46.

In the Advisory Action mailed on December 7, 2006 (Paper No.6), the Examiner alleges that “Dunlavey discloses grouping of like variables into a multidimensional variable including various components, with certain distinct like variables within this set representing non-interfaced parameters that are executed thereby becoming non-interfaced run-time parameters. See column 29, lines 17-21.” (See continuation sheet). Appellant respectfully disagrees and contends that the Examiner’s interpretation of Dunlavey is improper and the Examiner’s allegation is not supported by Dunlavey.

In the section referenced by the Examiner, Dunlavey states that “each unit category includes a plurality of unit names, each multidimensional unit type further comprising a conversion factor for conversion of a data value to a set of default units, each default unit selected from a unit category.” The multidimensional unit type of Dunlavey includes a plurality of unit categories, such as volume, weight, time, quantity, and age. (See column 16, lines 52-54). For example, the volumn category may include liter (L), deciliter (dL), centiliter(cL), milliliter(mL), nonoliter (nL), etc. (See column 16, lines 65-67). The

multidimensional unit type includes a default unit in each category, for example, liter (L) in the volume category. (See column 17, lines 32-34). The multidimensional unit type represents a unit for a variable with the plurality of default units. (See column 29, lines 10-15). The multidimensional unit type, however, does not include like variables grouped into a set, as alleged by the Examiner. Nor, as discussed above, does the multidimensional unit type correspond to a run-time parameter. The default units of the multidimensional variable are not like non-interfaced run-time parameters, as required in claims 33 and 46.

2. Dunlavey fails to disclose pooling “to reuse data for
The like non-interfaced run-time block parameters together”

Furthermore, Dunlavey fails to disclose that the pooling in claims 33 and 46 is conducted “to reuse data for the like non-interfaced run-time block parameters.”

The Examiner alleges that “Dunlavey discloses multidimensional data types that represent various like variables, which are then reused in relation to expressions and statements (column 3, lines 3-11).” (See Office Action of September 8, 2006, page 3). Appellant respectfully disagrees.

In the referenced section, Dunlavey discusses the use of the multidimensional unit type data to represent the units-specifying data in an internal format. The multidimensional unit type is used to represent a unit for a variable. That is, the multidimensional unit type of Dunlavey is merely an internal data structure for holding one parameter, where that parameter may be represented in a number of units. Dunlavey, however, does not disclose reusing data for pooled like parameters or variables. Appellant therefore submits that Dunlavey fails to disclose “pooling together like non-interfaced run-time block parameters to reuse data for the like non-interfaced run-time block parameters,” as required by claims

33 and 46. Dunlavey is silent about reusing parameter data to represent pooled like parameters.

In the ‘Response to Arguments’ section of the Office Action of September 8, 2006, the Examiner alleges that “Dunlavey also discloses multivariate distribution blocks, which hold like parameters. The variables are reused within various expressions, the distribution block being reused multiple times (column 3, lines 8-11).” (See Office Action of September 8, 2006, page 10). Appellant respectfully disagrees. Appellant also notes that the cited section of Dunlavey does not mention any of the multivariate distribution blocks, reusing variables within various expressions, or the distribution block being reused multiple times. The discussion below attempts to address the Examiner’s argument to the best of the Appellant’s understanding of what the Examiner has meant to refer to.

The multivariate distribution block of Dunlavey represents quantities that are known to be variable over time. (See column 8, lines 10-11). In contrast, claims 33 and 46 are directed towards operations on non-interfaced parameters – that is, parameters whose values do not change over the course of the execution. That is, even if the multivariate distribution block of Dunlavey were to be likened to the run-time parameters of the present application, the multivariate distribution block would correspond to interfaced parameters, and not to non-interfaced parameters, as claimed in claims 33 and 46. Moreover, the Dunlavey reference does not disclose reusing of data for the multivariate distribution block. The Dunlavey reference only discloses that the categorical variables can be jointly distributed, as discussed above with reference to Figure 2B of Dunlavey. The joint distribution does not relate to the reusing of parameter data, it only relates to the probabilistic characteristics of the data stored in the corresponding variables.

Appellant also respectfully disagrees with the allegation that the reuse of the multivariate distribution block corresponds to the reuse of data for the like non-

interfaced run-time block parameters. (See Office Action of September 8, 2006, page 10). The reuse of a block in a model is different from the reuse of block parameter data in an executable model or generated code. There may be any number of similar or identical blocks in the model, but that itself is in no way indicative as to what happens to their parameters during run-time. It would be incorrect and contrary to all that is stated in the Specification of the present application to read reusing non-interfaced run-time block parameters as the same as reusing blocks in a model. Appellant therefore again submits that Dunlavey does not disclose “pooling together like non-interfaced run-time block parameters to reuse data for the like non-interfaced run-time block parameters,” as required by claims 33 and 46.

For the reasons set forth above, Appellant respectfully requests that the 35 U.S.C. §102(e) rejection of claims 33-44 and 46 be reversed and claims 33-44 and 46 be allowed.

VIII. CLAIMS APPENDIX

A copy of the claims involved in the present appeal is attached hereto as Appendix A.

IX. EVIDENCE APPENDIX

No evidence pursuant to §§ 1.130, 1.131, or 1.132 or entered by or relied upon by the Examiner is being submitted.

X. RELATED PROCEEDINGS APPENDIX

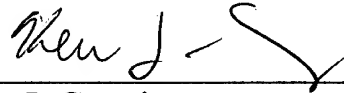
No copies of decisions of related proceedings referenced in II are being provided.

Application No.: 09/911,663

Docket No.: MWS-072

Dated: October 12, 2007

Respectfully submitted,

By  _____

Kevin J. Canning

Registration No.: 35,470

LAHIVE & COCKFIELD, LLP

One Post Office Square

Boston, Massachusetts 02109-2127

(617) 227-7400

(617) 742-4214 (Fax)

Attorney/Agent For Appellant

APPENDIX A

Claims Involved in the Appeal of Application Serial No. 09/911,663

33. A method of mapping graphical block diagram block parameters in a graphical block diagram modeling environment, comprising:
- receiving a plurality of user-defined block parameters;
 - processing the plurality of user-defined block parameters to produce a plurality of run-time block parameters;
 - pooling together like non-interfaced run-time block parameters to reuse data for the like non-interfaced run-time block parameters.
34. The method of claim 33, wherein pooling further comprises mapping user-defined block parameters into an existing pool.
35. The method of claim 33, wherein the non-interfaced run-time block parameters have stored values that differ from presented values.
36. The method of claim 35, wherein the non-interfaced run-time block parameters are fixed point.
37. The method of claim 33, further comprising translating at run-time constant parameter values to an internal representation to enable increased pooling.
38. The method of claim 33, wherein the step of pooling further comprises collecting constant portions of an expression containing an interfaced variable.

39. The method of claim 33, wherein the run-time block parameters are configured to return at least one of simulation results, and automatically generated code that implements graphical block diagram model equations.
40. The method of claim 39, wherein when the code is automatically generated, parameter expressions are maintained in the automatically generated code.
41. The method of claim 40, wherein the parameter expressions contain interfaced variables which are updatable.
42. The method of claim 41, further comprising converting to a relatively more compact representation portions of the parameter expressions that are constants while providing access to interfaced variables.
43. The method of claim 41, wherein interfaced variables are updatable.
44. The method of claim 43, wherein updatable variables used in a plurality of blocks are declared only once.
46. A medium for use in a graphical modeling environment on an electronic device, the medium holding instructions executable using the electronic device for performing a method of mapping graphical block diagram block parameters, the method comprising:
- receiving a plurality of user-defined block parameters;
 - processing the plurality of user-defined block parameters to produce a plurality of run-time block parameters; and

pooling together like non-interfaced run-time block parameters to reuse data for the like non-interfaced run-time block parameters.